Federated Learning & Data Privacy, 2024-2025
Nguyen Dinh Huy

Third Lab - 11 February 2025

Welcome to the third lab session of the Federated Learning & Data Privacy course! Today we will see how to implement Federated Learning in real network systems.

I. EXERCISE 5 - Federated Learning with Flower

Objective: Gain practical experience with the Flower federated learning framework by deploying a real, distributed federated learning experiment. Explore personalized federated learning algorithms and compare their performance against the standard FedAvg.

II. EXERCISE 5.1 - Federated Learning on Real Networks

Goal: Deploy a federated learning system using PyTorch and Flower to understand the setup and execution of federated learning in a networked environment.

Setup

1. **Clone the Flower repository and set up the PyTorch quickstart example**:

```
git clone --depth=1 https://github.com/adap/flower.git && mv flower/examples/quickstart-pytorch . && rm -rf flower && cd quickstart-pytorch
```

2. **Install the dependencies**:

```
pip install -e .
```

Run Federated Learning

- **Launch the simulation**

```
flwr run .
```

- Observe the federated training process initiated by PyTorch through Flower.

You can find complete instructions here.

III. EXERCISE 5.2 - Tackling Data Heterogeneity with FedProx

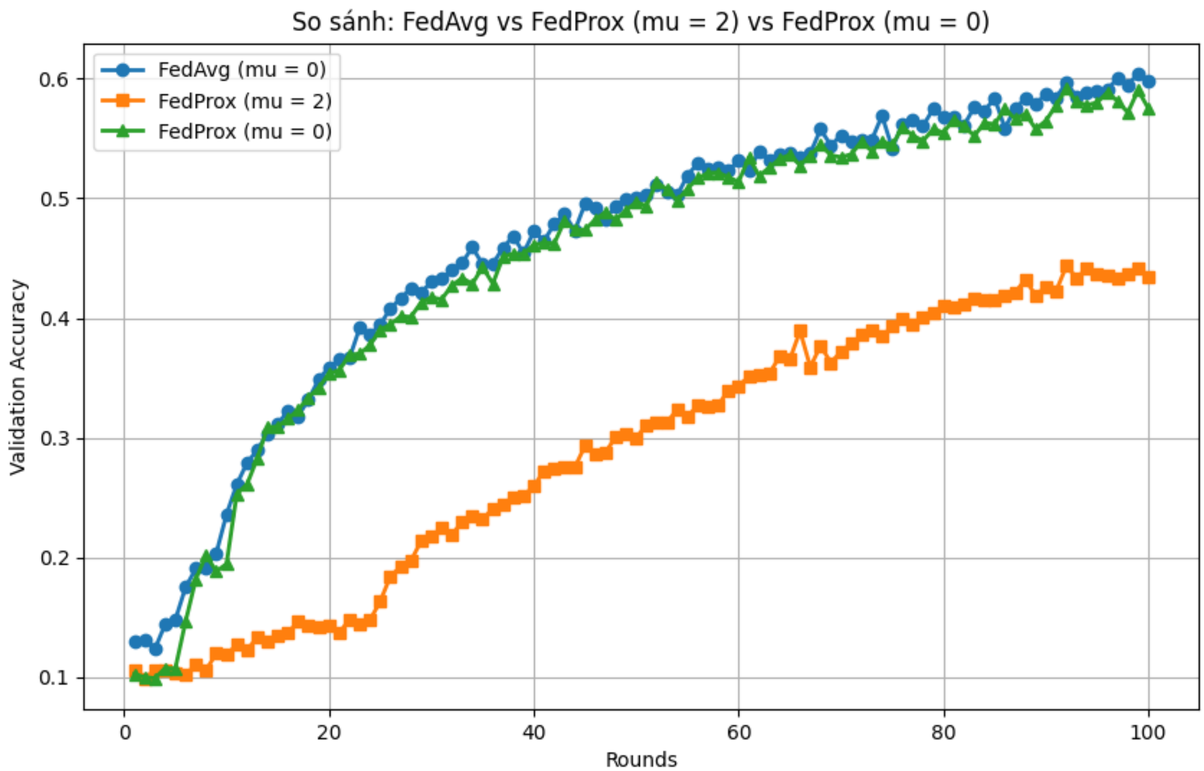
Objective: Understand how the FedProx algorithm addresses the challenges posed by data heterogeneity in federated learning and compare its performance with the FedAvg algorithm.

- **FedProx Overview**: FedProx modifies the local training objective by introducing a proximal term, which aims to reduce local model drift by penalizing significant deviations from the global model. Review the FedProx algorithm Federated Optimization in Heterogeneous Networks (Algorithm 2).
- **Instructions**: Follow the tutorial on FedProx available at Flower's documentation. Run experiments trying different number of rounds and different mu values.

- **Analysis**: Discuss the observed differences in performance between FedAvg and FedProx. Are there specific configurations (e.g., number of local epochs) where FedProx particularly outperforms FedAvg?

RESULT :

Code Link [: https://github.com/DinhHuy1405/Federated-Learning-2024---2025/tree/main/TP3](https://github.com/DinhHuy1405/Federated-Learning-2024---2025/tree/main/TP3)



Observations from the Experiment

The graph shows validation accuracy over **100 rounds** for three federated learning approaches:

1. **FedAvg (μ = 0)** → Traditional federated averaging.
2. **FedProx (μ = 2)** → FedProx with a strong regularization term.
3. **FedProx (μ = 0)** → Equivalent to FedAvg.

Key Insights:

1. FedAvg (μ = 0) vs. FedProx (μ = 0):

- As expected, **FedAvg (μ = 0)** and **FedProx (μ = 0)** behave similarly because FedProx with μ = 0 is equivalent to FedAvg.
- There are slight performance variations, but both approaches show almost identical trends in accuracy improvement over time.
- Any small differences could be attributed to implementation details or stochastic effects.

2. FedAvg (μ = 0) vs. FedProx (μ = 2):

- **FedAvg (μ = 0)** achieves significantly **higher accuracy** over time compared to **FedProx (μ = 2)**.
- **FedProx (μ = 2) starts with a similar accuracy but converges much more slowly**.
- This suggests that a strong proximal term (μ = 2) effectively **reduces client drift** but also **restricts adaptation**, leading to **lower overall accuracy**.

3. FedProx (μ = 2) vs. FedProx (μ = 0):

- **FedProx (μ = 0) consistently outperforms FedProx (μ = 2)**.
- This indicates that for this dataset and training setup, a **strong proximal constraint (μ = 2) overly restricts local model updates**, preventing optimal learning.

Analysis of Data Heterogeneity

- **FedProx** is designed to mitigate **data heterogeneity** by limiting client model divergence.
- However, selecting an appropriate **μ** value is critical:
  - **Small μ (close to 0)**: Allows more flexibility but may suffer from client drift.
  - **Large μ** : Reduces drift but slows down convergence, leading to suboptimal performance.

Impact of Local Epochs

- The impact of FedProx **may vary depending on the number of local epochs**:
  - **If local training lasts multiple epochs**, FedAvg may diverge more due to heterogeneous data distributions.
  - **FedProx can help by constraining local updates**, keeping them closer to the global model.
  - **In scenarios with higher local epochs, FedProx might perform better** than FedAvg.

IV. BONUS EXERCISE - Personalized Federated Learning

Goal: Evaluate a personalized federated learning algorithm using Flower, showing its possible advantages over the FedAvg algorithm.

Choose one of the two proposed personalization algorithms:

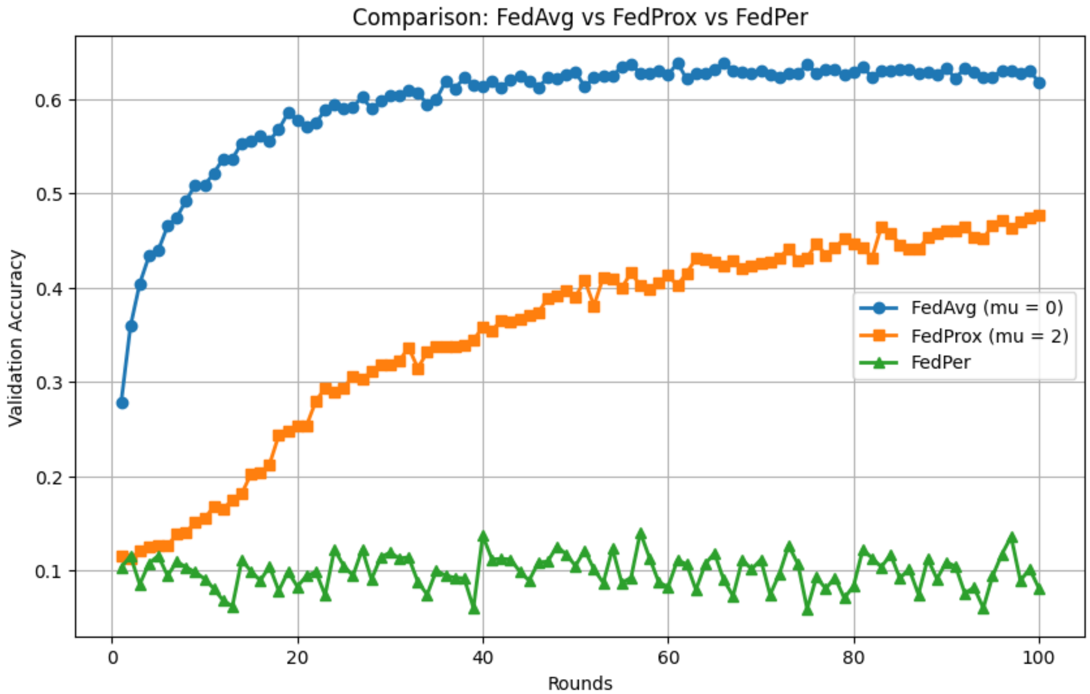
1. **Federated Learning with Personalization Layers (FedPer)**
   - **Overview**: FedPer implements personalization by allowing some neural network layers to be client-specific, making the model closer to individual data distributions.
   - **Instructions**: Follow the tutorial on FedPer available at [Flower's documentation].

Evaluation

- Reproduce the tutorial and compare the results with FedAvg to highlight the benefits of personalization.

RESULT:

Code Link: https://github.com/DinhHuy1405/Federated-Learning-2024---2025/tree/main/TP3



Observations from the Experiment

The graph displays the validation accuracy over **100 rounds** for three federated learning strategies:

1. **FedAvg (μ = 0)** → Traditional federated averaging.
2. **FedProx (μ = 2)** → FedProx with a strong regularization term.
3. **FedPer** → An approach focusing on personalized federated learning.

Key Insights:

1. FedAvg (μ = 0):

- **FedAvg (μ = 0)** shows the highest and most stable performance, achieving over **60% accuracy**.
- It is the most effective strategy among the three, displaying rapid and consistent improvements in accuracy.

2. FedProx (μ = 2):

- **FedProx (μ = 2)** starts with lower accuracy but gradually improves, stabilizing around **40% accuracy**.
- The strong regularization parameter (μ = 2) seems to limit performance initially but ensures moderate improvements over time.

3. FedPer:

- **FedPer**, aiming at personalized models, shows much lower and fluctuating performance, with accuracy hovering around **10%**.
- This suggests potential issues in personalization effectiveness or instability in the learning process under this specific setup.

Comparison and Analysis:

- **FedAvg (μ = 0) vs. FedProx (μ = 2):**
  - As seen before, **FedAvg** consistently outperforms **FedProx with a strong μ**. The lack of a regularization constraint allows for more adaptable but possibly more varied client updates.
  - **FedProx** reduces client drift but at a significant cost to convergence speed and ultimate performance.
- **FedAvg (μ = 0) vs. FedPer:**
  - **FedAvg** significantly outperforms **FedPer**, indicating that the non-personalized model is more effective in this scenario.
  - **FedPer's** performance suggests that its approach to personalization might not be effectively capturing useful patterns or it might be too sensitive to client-specific noise.
- **FedProx (μ = 2) vs. FedPer:**
  - Even with its conservative updates, **FedProx** performs better than **FedPer**, further highlighting the challenges in the personalization strategy employed by **FedPer**.